Chapter 1 : Talking to Tim O'Reilly about the Architecture of Participation - O'Reilly FYI Blog

Free Programming Ebooks. We've compiled the best insights from subject matter experts and industry insiders for you in one place, so you can dive deep into the latest of what's happening in the world of software engineering, architecture, and open source.

In this excerpt from Beautiful Teams: We wanted to talk to Tim to hear his ideas about how he built his team and continues to bring out the best in them. One thing we noticed when we started putting together this book was that everyone seems to have a slightly different definition of "team. We kind of pulled a fast one and just left it open to interpretation. So how do you define a team? My own experiences are around running a company. Yes, there are team experiences in that. But most of the reflections I have are around the broader question of how you exert leadership. Let me start at the top, with a few thoughts about leadership and management, which are part of the whole team thing. There are two quotes I want to start you off with. One is from Harold Geneen, who was the guy who started ITT, which was really the first modern conglomerate. And he said, "The skill of management is achieving your objectives through the efforts of others. And the question is of the different styles of doing that, where some of them are very directive. This is the classic "manager"--the idea of somebody who figures out what needs to be done, and who needs to do it, and builds the teams with the roles, and so on. While I completely subscribe to the concept, because the skill of management is indeed achieving your objectives through the effort of others, I have always worked with the framing of another quote, which is actually about writing. He said, "The skill of writing is to create a context in which other people can think. So, can you think of how you apply that idea--"creating a context in which other people can think"--to software teams? In , we did a book called Open Sources, and we did interviews for some of the people who produced their essays on open source. Because there are rules that are laid down. We have to have a system that has a fundamental characteristic that there are small pieces that people can work on independently. Because a book is a fairly large, complex thing with a single narrative thread. Wikipedia is a set of pages, and the atomic unit of content is something that a single individual can make a plausible promise at, and other people can update and tweak. And the whole is the sum of many, many such small parts. Unix is designed a little more like a set of LEGOs, where the design principles are that you have these "innies" and "outies" and they snap together. Thinking about pipes and filters and all of those kinds of things--that people can write completely independent utilities with the knowledge that they just fit. Most programs read standard in and wrote standard out, and there were a few simple rules. Similarly, every atomic piece has a manpage. Let me give you a concrete story. Back in , when I organized what came to be called the Open Source Summit, I had this plausible story about what was happening in the industry. We were all thinking about a bunch of things in parallel. Netscape had just open sourced their browser, so there was a lot of ferment. And I noticed something: Meanwhile, there was this whole other tradition that I looked at. And I looked at it and said, "Wow, that tradition has actually had more impact. Number two, either sendmail or Apache. Seventy-five percent of Net email is routed by sendmail. It sounds like you found a way to rise above that and bring everyone together. How did that happen? I brought together all the GPL people and all the Berkeley people, and I said, "We have to find out what we have in common. We have to tell a story. We had the press conference at five, and the rest is history. It seems like it really helped you guys. A lot of what happened was that we brought together a very short-term group of people. But I knew I wanted to make something happen. Now, a lot of companies and a lot of people really misuse that. But when you do it right, identifying that sense of urgency can be really good. In terms of teams, the other thing that I would just really observe is the power of having people with complementary strengths. Understanding complementary strengths is really critical. Practices like adopting a very strict build and release process, continuous integration, test-driven development. Lots and lots of code reviews. So why do you think that is? Well, let me put it this way. Lao Tzu said this 2, years ago: When some people create something really wonderful with an aesthetic vision, it seems really obvious. Now that we have the iPod, how could you imagine not having touch-screen devices? I remember the first time I picked up the Kindle. Well, you have to realize that people always get hung up on licensing as

critical to open source. And while it is certainly important, I think as you guys have suggested here, the practices are much, much more important. We have to put out a plausible idea of something we want to get done. Especially not in Perl. And take the book Programming Perl. So I blessed it and sent it through. I remember this one book we published. I said, "All you have to do now is tell the story of the book so that this makes sense. You had an expectation of what the book was going to be, but what you were given turned out to be different--and better. I think this is almost the root of intelligence that draws everything else. There are different kinds of intelligence. One kind is essentially algorithmic and manipulative: And the smartest people have both of those qualities. They can look at the world fresh; they can look at something and say, "Wow, I see what this can be. And that got in the way of her being able to see the material with fresh eyes and understand that the process could be better by just helping the authors to do what they wanted anyway. Do you think that an editor--or any team leader, really--should try not to have a strong hand in pushing the authors or the team toward a specific goal? Most of my experience is very much being a leader, not necessarily being a team member. A great deal of it comes from directing people, and I try to direct them in such a way that I have as little to do as possible. I think editing a book is like that. Leading a project is like that. When I started telling the story about Web 2. Same thing when I told the story about open source. I think that leading your team is like that also. How do you get a group of people to achieve their potential? By seeing who they are, and what they can accomplish. That has more of a collective leadership? Yes, it is possible. Take Apache, because I think Apache is the great example of that. Tim Berners-Lee laid down the blueprint. What we do is a hypertext server, and we have this nice extension mechanism where people who want to do something else can add it on. They kept to a pure vision. Apache came from a group of people who were abandoned by the NCSA server team when they all went to found Netscape. And there were a bunch of customers, so they said, "We have to maintain this, and keep it going. There were some wonderful principles laid down, and people really honored them. The point is that if you have the system architected right, you have a better chance of success for teams. Maybe a disaster or two that maybe you learned something important from? There are probably quite a few disasters over the years, some of them which were turned into successes after the fact. Things can go wrong. Look at what happened here with Yahoo! But we might end up looking back and saying, "Wow, Jerry was brilliant. What if IBM had been the sharp dealer? Right, what we think of as a set-in-stone success now might not even have seemed like a wise choice at the time. You make a series of choices, and the chips fall where they may.

Chapter 2 : Programming Interactivity, 2nd Edition - O'Reilly Media

O'Reilly is a learning company that helps individuals, teams, and enterprises build skills to succeed in a world defined by technology-driven transformation. From in-person conferences and live online training courses to self-directed learning and immediate access to problem solving online, O'Reilly has you and your team covered.

Chapter 3 : 8 Android Programming Most Popular O'Reilly Books - Best Programming Books

Over 40, books, videos, and interactive tutorials from over of the world's best publishers, including O'Reilly, Pearson, HBR, and Packt.

Chapter 4 : 9 R Most Popular O'Reilly Books - Best Programming Books

Online shopping from a great selection at Books Store.

Chapter 5 : Free Programming Ebooks - O'Reilly Media

The following 9 R books are the current best-sellers of O'Reilly's publishing company.

DOWNLOAD PDF O REILLY PROGRAMMING BOOKS

Chapter 6 : O'Reilly Media - Technology and Business Training

And then it struck us that you might want to complement that course with some of the 36 free ebooks on computer programming from O'Reilly Media--of which 7 are dedicated to Python itself. Other books focus on Java, C++, Swift, Software Architecture, and more.

Chapter 7 : O'Reilly Media - Wikipedia

With this hands-on guide, you'll explore several themes in interactive art and designâ€”including 3D graphics, sound, physical interaction, computer vision, and geolocationâ€”and learn the basic programming and electronics concepts you need to implement them.

Chapter 8 : 36 eBooks on Computer Programming from Oâ€™Reilly Media: Free to Download and Read |

of over 3, results for "o'reilly programming books" Version Control with Git: Powerful tools and techniques for collaborative software development.

Chapter 9 : O'Reilly E-books - Computer Programming - InfoGuides at Rochester Institute of Technology

/r/programming is a reddit for discussion and news about computer programming. Guidelines. Please keep submissions on topic and of high quality. Just because it has a computer in it doesn't make it programming.